Project 2

Daniel Fredin, Junhan Li, & Eric Chen

Introduction

By utilizing the glass identification dataset graciously provided by the USA Forensic Science Service, we have successfully derived insights and constructed a regression equation to forecast the occurrence of silicon (Si). This prediction is based on the observed oxide content found in the various types of glass examined in the study.

Problem 1

Part(a)

```r
# Correlation for all numerical variables
cor(glass[,c("Si", "RI", "Na", "Mg", "Al", "K", "Ca", "Ba", "Fe")])
```

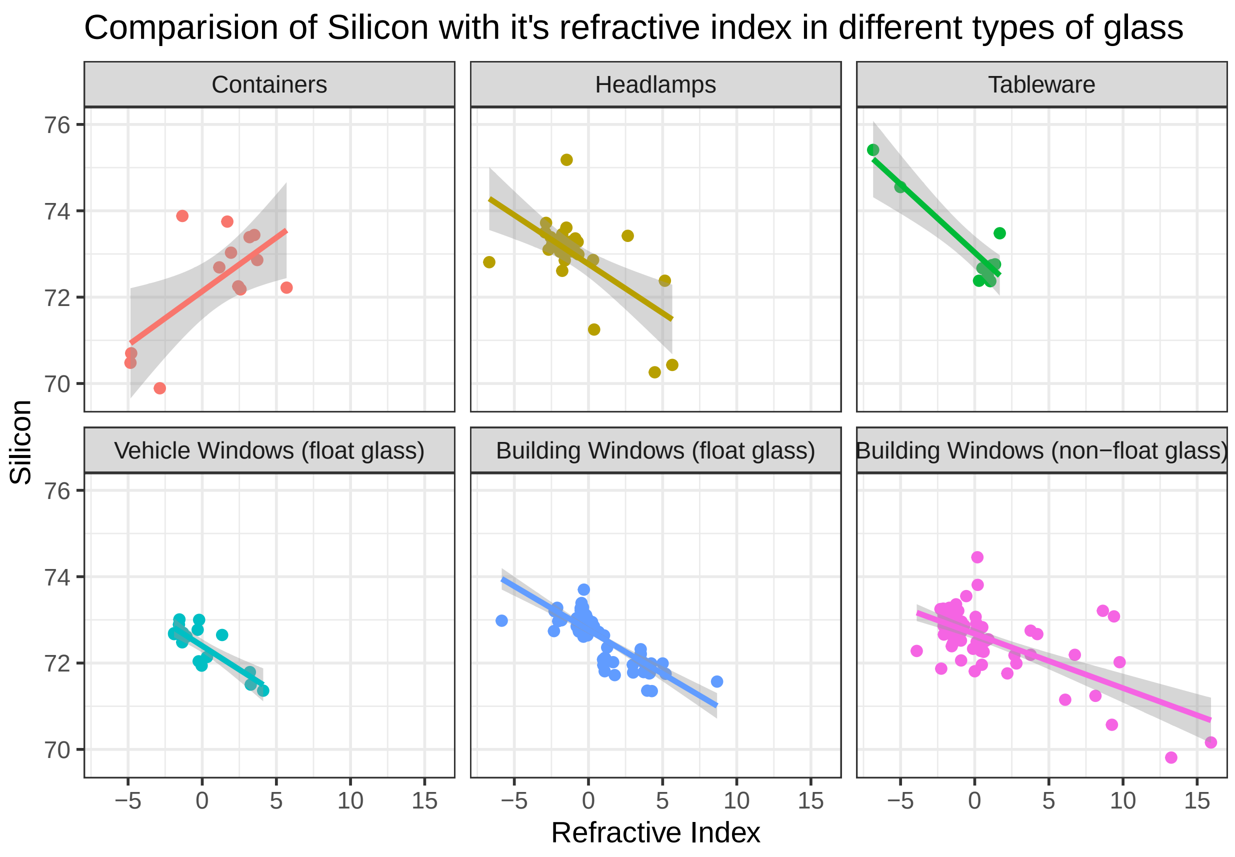
```
##             Si            RI          Na           Mg          Al            K
## Si  1.00000000 -0.5420521997 -0.06980881 -0.165926723 -0.00552372 -0.193330854
## RI -0.54205220  1.0000000000 -0.19188538 -0.122274039 -0.40732603 -0.289832711
## Na -0.06980881 -0.1918853790  1.00000000 -0.273731961  0.15679367 -0.266086504
## Mg -0.16592672 -0.1222740393 -0.27373196  1.000000000 -0.48179851  0.005395667
## Al -0.00552372 -0.4073260341  0.15679367 -0.481798509  1.00000000  0.325958446
## K  -0.19333085 -0.2898327111 -0.26608650  0.005395667  0.32595845  1.000000000
## Ca -0.20873215  0.8104026963 -0.27544249 -0.443750026 -0.25959201 -0.317836155
## Ba -0.10215131 -0.0003860189  0.32660288 -0.492262118  0.47940390 -0.042618059
## Fe -0.09420073  0.1430096093 -0.24134641  0.083059529 -0.07440215 -0.007719049
##            Ca            Ba           Fe
## Si -0.2087322 -0.1021513105 -0.094200731
## RI  0.8104027 -0.0003860189  0.143009609
## Na -0.2754425  0.3266028795 -0.241346411
## Mg -0.4437500 -0.4922621178  0.083059529
## Al -0.2595920  0.4794039017 -0.074402151
## K  -0.3178362 -0.0426180594 -0.007719049
## Ca  1.0000000 -0.1128409671  0.124968219
## Ba -0.1128410  1.0000000000 -0.058691755
## Fe  0.1249682 -0.0586917554  1.000000000
```

```r
# Select the highest correlated variables
glass_new <- glass %>%
  select(Si, RI, type)

# Rename for facet wrap labels
```

```
glass_names <- as_labeller(
  c('Con' = "Containers",
    'Head' = "Headlamps",
    'Tabl' = "Tableware",
    'Veh' = "Vehicle Windows (float glass)",
    'WinF' = "Building Windows (float glass)",
    'WinNF' = "Building Windows (non-float glass)")
)

# Scatter plot of Si vs RI for each glass type
ggplot(data= glass_new, aes(x=RI, y= Si, color = type)) +
  geom_point() +
  geom_smooth(method = lm, se = T) +
  facet_wrap(~type, labeller = glass_names) +
  labs(title = "Comparision of Silicon with it's refractive index in different types of glass",
       x = "Refractive Index",
       y = "Silicon",
       color = "Glass Types") +
  theme_bw() +
  theme(legend.position = "none") +
  guides(color = guide_legend(nrow = 1))
```



To visualize the relationship between two numerical variables and illustrate the variation across different types of glass, we conducted a correlation analysis. Our aim was to identify the pair of variables with the highest correlation coefficient. In accordance with the given problem, we selected silicon (Si) as our dependent variable. Subsequently, we chose the refractive index (RI) as the independent variable due to its strong correlation with Si, indicated by a coefficient of -0.54. This implies a moderately negative correlation between Si and RI. As a result, we can infer that as the silicon content decreases, the refractive index of the glass is expected to increase.

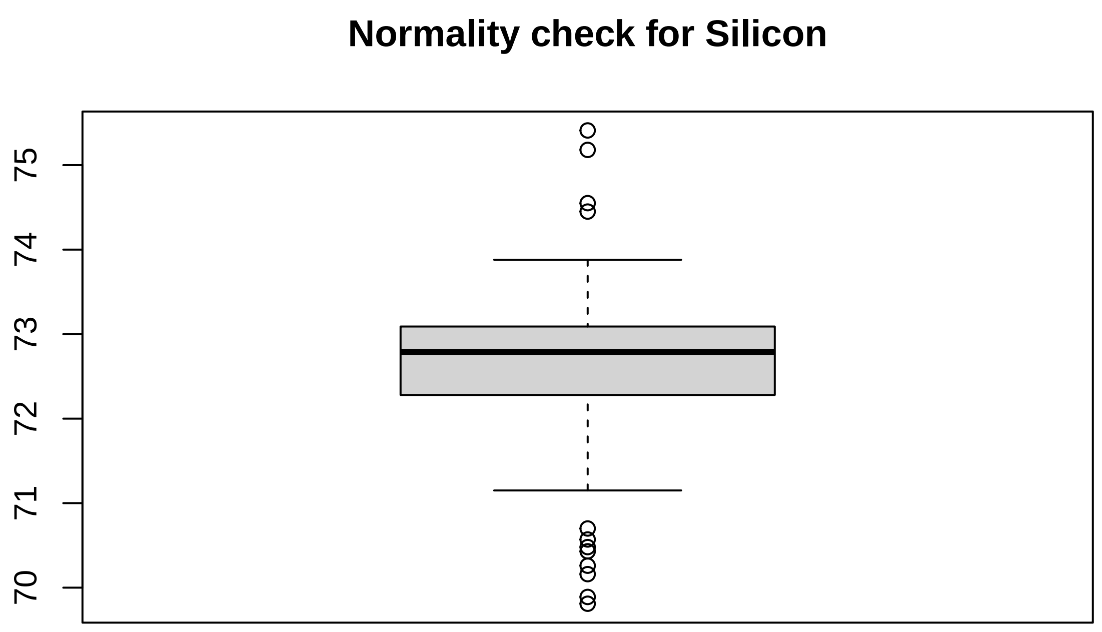
Based on our visualization, it is evident that there exists a moderately negative correlation between Si and RI across all glass types, except for container glass, which surprisingly exhibits a moderately positive correlation. This peculiar pattern may be attributed to the elevated presence of water-insoluble oxides in container glass, contributing to enhanced chemical durability against water, a crucial requirement for storing beverages and food. Another noteworthy observation is that the refractive index tends to cluster around 0 for all glass types. This indicates that light passing through the glass travels at a velocity equal to that in a vacuum, exhibiting no bending or refraction. Such consistency around the refractive index of 0 appears to be a desirable characteristic across all glass types. While the majority of glass types exhibit a RI range between -5 and 5, it is worth noting that building windows display the widest RI range, with float glass spanning from -5 to 10 and non-float glass ranging from -5 to 15. On the other hand, vehicle glass demonstrates the smallest RI range, spreading from -2 to 4. In terms of Si concentration, tableware stands out with the highest Si content compared to other glass types, whereas building windows (non-float glass) generally possess the lowest Si concentration. Additionally, tableware showcases the most significant decline in Si per unit decrease in RI, indicating a steep relationship between these variables.

Part (b)

```
# We will first check for the normality of dependent variable
shapiro.test(glass$Si)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  glass$Si
## W = 0.91966, p-value = 2.175e-09
```

```
boxplot(glass$Si, main = "Normality check for Silicon")
```



```r
set.seed(123)

# Regular linear regression with all variables
glass_model1 <- lm(Si ~., data = glass)
predictions <- glass_model1 %>% predict(glass)
model1_results <- data.frame( R2 = R2(predictions, glass$Si),
            RMSE = RMSE(predictions, glass$Si),
            MAE = MAE(predictions, glass$Si))

# Regular linear regression with 3 variables from our visualization
glass_model2 <- lm(Si  ~ RI + factor(type), data = glass)
predictions <- glass_model2 %>% predict(glass)
model2_results <- data.frame( R2 = R2(predictions, glass$Si),
             RMSE = RMSE(predictions, glass$Si),
             MAE = MAE(predictions, glass$Si))

# Training and testing method
training <- glass$Si %>%
    createDataPartition(p = .75, list = FALSE)
train.data  <- glass[training, ]
test.data <- glass[-training, ]

glass_model3 <- lm(Si ~ Na + Mg + Al + K + Ca + Ba + Fe, data = train.data)
predictions <- glass_model3 %>% predict(test.data)
model3_results <- data.frame( R2 = R2(predictions, test.data$Si),
            RMSE = RMSE(predictions, test.data$Si),
            MAE = MAE(predictions, test.data$Si))
```

Table 1: Summary of the accuracy and error exhibited by our three models.

	R2	RMSE	MAE
Model 1	0.9849829	0.0946942	0.0663155
Model 2	0.3285257	0.6332055	0.4205404
Model 3	0.9930191	0.0749413	0.0608737

To select our best model, we first created a model using all the variables except Si. Our second model was selected based on the variables we chose for our visualization, as we saw noticeable correlation between the RI and Si content of glass. The final model had all the significant predictor variables from model 1, which happened to be all the other elements. We also used the training and testing method to create model 3 with 75% training data and 25% testing data.

We chose Model 3 as our best model since it had the highest R^2 and the lowest RMSE. The R^2 value of 0.993 suggests that 99.3% of the variation of Si in our data could be explained by the independent variables while the RMSE value of 0.075 is lower than those of our other two models. This lower prediction error means a higher accuracy of prediction on the test data and thus a model that could more effectively predict the percentage of Si in glass with low margins of error.

From the Shapiro Test on the Si data, we obtained a p value significantly less than 0.05. This suggests a non-normal distribution of Si. Before creating our models, we attempted multiple types of transformations, including square root, logarithmic, and boxcox. While some of these transformations raised the p value of the Shapiro Test, none were successful in creating normally-distributed data. However, since a histogram of the data appears as a bell curve, and the box plot of the data appears normal as well, we decided to create our models despite this inconvenience, and the best model we selected yields great predictive power regardless.

```r
# Optimization of our best model
min.RSS <- function(data, par) {
              with(data, sum((par[1] + par[2] * Na + par[3] * Mg + par[4] * Al
                              + par[5] * K + par[6] * Ca
                              + par[7] * Ba + par[8] * Fe - Si)^2))
}

result <- optim(par = c(98, -1, -1, -1, -1, -1, -1, 0.5),
                fn = min.RSS,
                data = train.data)
```

Table 2: Summary of the coeffcients between our best model and our optimized model.

	(Intercept)	Na	Mg	Al	K	Ca	Ba	Fe
Model 3	98.81728	-0.9418716	-0.9921217	-1.003616	-0.9800752	-0.9769767	-0.960229	-0.349477
Optimized Model 3	99.37793	-0.9786847	-0.9982519	-1.003050	-1.0036449	-0.9808882	-0.961868	-0.379274

```r
# Fit the final model with the optimized coefficients
final_model <- lm(Si ~ Na + Mg + Al + K + Ca + Ba + Fe, data = train.data)
```

```r
# Extract the optimized coefficients
optimized_coefficients <- result$par

names(optimized_coefficients) <- c("(Intercept)" ,"Na","Mg","Al","K","Ca","Ba","Fe")
# Update the coefficients of the final model with the optimized coefficients
final_model$coefficients <- optimized_coefficients

# Test final model
predictions <- final_model %>% predict(test.data)
final_model_results <- data.frame( R2 = R2(predictions, test.data$Si),
            RMSE = RMSE(predictions, test.data$Si),
            MAE = MAE(predictions, test.data$Si))
```

Table 3: Summary of the accuracy and error exhibited by all our models.

	R2	RMSE	MAE
Model 1	0.9849829	0.0946942	0.0663155
Model 2	0.3285257	0.6332055	0.4205404
Model 3	0.9930191	0.0749413	0.0608737
Optimized Model 3	0.9935043	0.0720492	0.0596898

We took our best model (Model 3) and used the coefficients from that model to create an optimized model with a slightly higher coefficient of determination and lower prediction error than our best non-optimized model (RMSE dropped from 0.075 to 0.072). Using the dependent variables from Model 3, the optimized model selected coefficients that minimizes the prediction error of Si from our training data.

Part (c)

- (i) Coefficient of determination

```r
R2 <- final_model_results[1]
R2
```

```
##          R2
## 1 0.9935043
```

The coefficient of determination, also known as R-squared, is a metric used in regression analysis to evaluate how well a model fits the data. It indicates the proportion of the dependent variable's variability that can be explained by the independent variables. R-squared ranges from 0 to 1, where 0 signifies that the independent variables have no influence on the variation, and 1 means they explain all of it. Our optimized best model had an impressive R-squared value of 0.9935,indicating that 99.35% of the dependent variable's variance is captured by the independent variables.

- (ii) Least-squares estimates for the regression line

```r
summary(final_model)
```

```
##
## Call:
## lm(formula = Si ~ Na + Mg + Al + K + Ca + Ba + Fe, data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.73472 -0.04143  0.01324  0.06600  0.20186
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 99.37793    0.35017 283.800  < 2e-16 ***
## Na          -0.97868    0.01604 -61.028  < 2e-16 ***
## Mg          -0.99825    0.01336 -74.740  < 2e-16 ***
## Al          -1.00305    0.02898 -34.617  < 2e-16 ***
## K           -1.00364    0.02047 -49.036  < 2e-16 ***
## Ca          -0.98089    0.01402 -69.968  < 2e-16 ***
## Ba          -0.96187    0.02492 -38.592  < 2e-16 ***
## Fe          -0.37927    0.09497  -3.993 0.000101 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1123 on 154 degrees of freedom
## Multiple R-squared:  0.9775, Adjusted R-squared:  0.9765
## F-statistic:   957 on 7 and 154 DF,  p-value: < 2.2e-16
```

Regression equation:

$$\begin{aligned} Si = 99.38 &- (0.98 \cdot Na) - (0.99 \cdot Mg) - (1.00 \cdot Al) \\ &- (1.00 \cdot K) - (0.98 \cdot Ca) - (0.96 \cdot Ba) - (0.38 \cdot Fe) \end{aligned}$$

- (iii) Slope and intercept interpretation

The regression coefficients correspond to the weight percent of each respective oxide. Thus, the intercept indicates that in the absence of any oxides, the glass will primarily consist of 99.38% silicon. The negative slopes indicate that an increase in the concentration of a specific oxide leads to a decrease in the silicon oxide content. Most of the oxide coefficients are approximately 1, except for iron oxide (Fe). For example, a one unit increase in Na oxide results in a 0.98 decrease in Si content, assuming all other factors remain constant. This relationship is sensible, as the addition of one unit of another oxide typically leads to a corresponding decrease in Si content. However, a one unit increase in Fe oxide content only causes a 0.38 decrease in Si content, while holding all other factors constant. This discrepancy may be attributed to the fact that Fe is an impurity in the glass-making process and is not intentionally added to achieve desired effects, such as Na or Ca.

At a 10% confidence level, all of the variables in our optimized best model exhibit significant predictability for Si content, as indicated by their p-values being less than 0.10. The variables found to be significant predictors are Na, Mg, Al, K, Ca, Ba, and Fe.

- (iv) Residual analysis

```r
# Check for normality of using shapiro test on residuals
# (This predicts residuals on training data)
real_resid <- residuals(final_model)
shapiro.test(real_resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  real_resid
## W = 0.8722, p-value = 1.541e-10
```

```
# (This predicts residuals on testing data)
resids <- test.data$Si - predictions
shapiro.test(resids)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resids
## W = 0.9818, p-value = 0.605
```

```
# Check for Linearity  using rainbow test
raintest(final_model)
```

```
##
##  Rainbow test
##
## data:  final_model
## Rain = 0.59126, df1 = 81, df2 = 73, p-value = 0.9892
```

```
# Check for multicollinearity
cat("Check for multicollinearity\n\n")
```

```
## Check for multicollinearity
```

```
vif(final_model)
```

```
##       Na       Mg       Al        K       Ca       Ba       Fe
## 1.857722 4.492997 2.718487 1.739089 3.400448 1.798373 1.071352
```

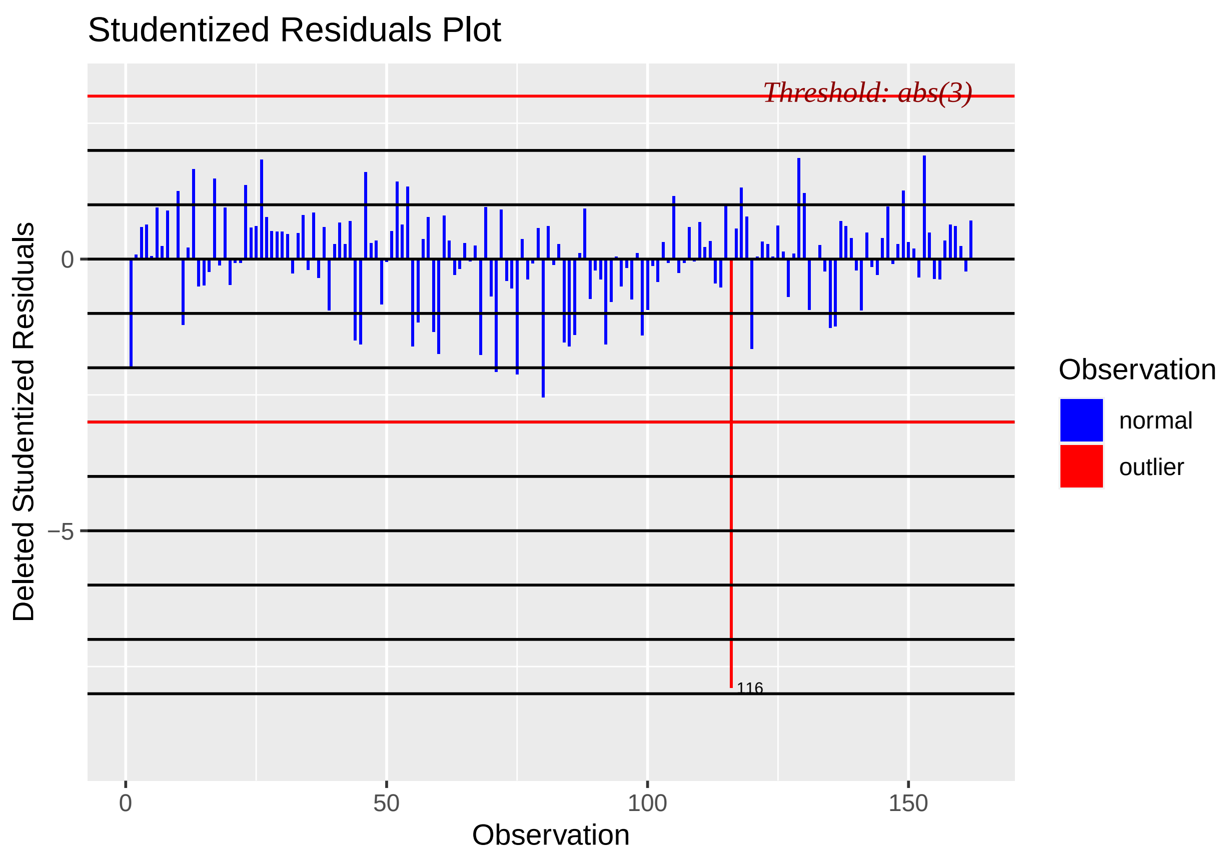
```
# Check for autocorrelation
dwtest(final_model)
```

```
##
##  Durbin-Watson test
##
## data:  final_model
## DW = 2.0254, p-value = 0.501
## alternative hypothesis: true autocorrelation is greater than 0
```

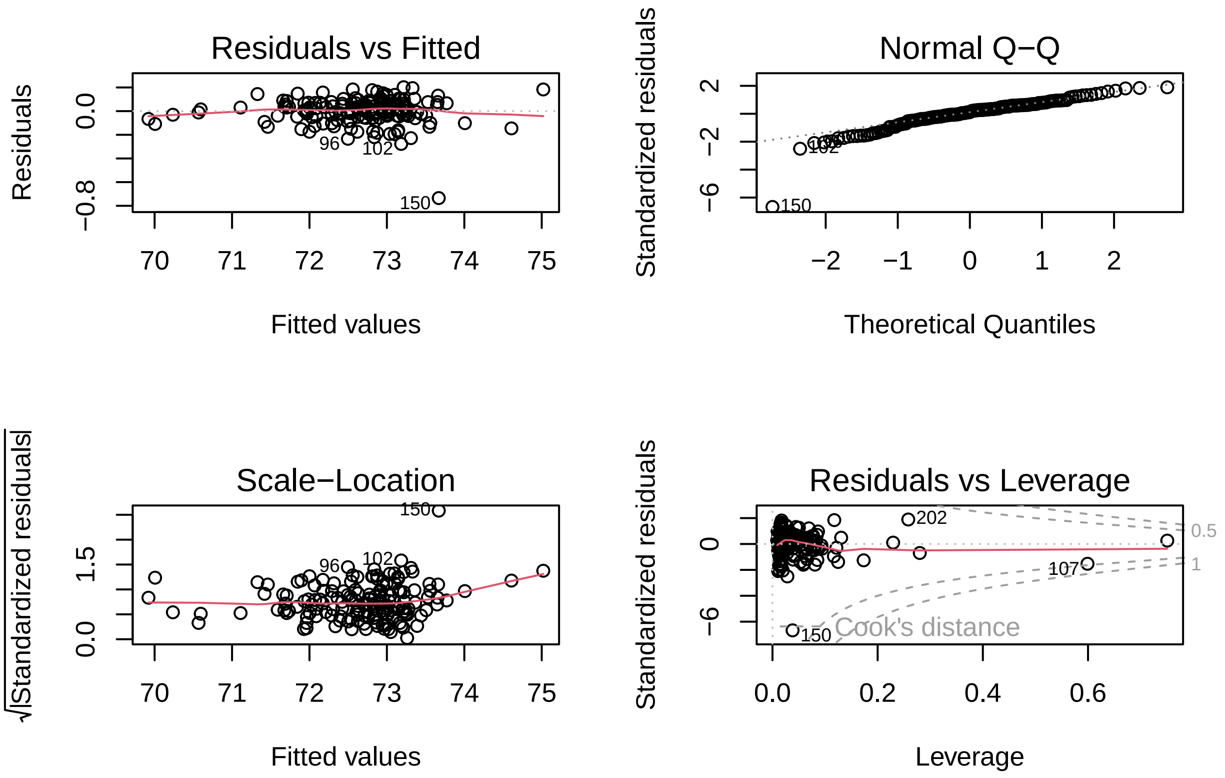
```
# Check for Heteroskedasticity
ols_test_breusch_pagan(final_model)
```

```
##
##   Breusch Pagan Test for Heteroskedasticity
##   -----------------------------------------
##   Ho: the variance is constant
##   Ha: the variance is not constant
##
##                Data
##   ------------------------------
##   Response : Si
##   Variables: fitted values of Si
##
##           Test Summary
##   ------------------------------
##   DF              =    1
##   Chi2            =    22.46384
##   Prob > Chi2     =    2.14137e-06
```

```
# Check for outliers
ols_plot_resid_stud(final_model)
```



```
par(mfrow = c(2,2))
plot(final_model)
```

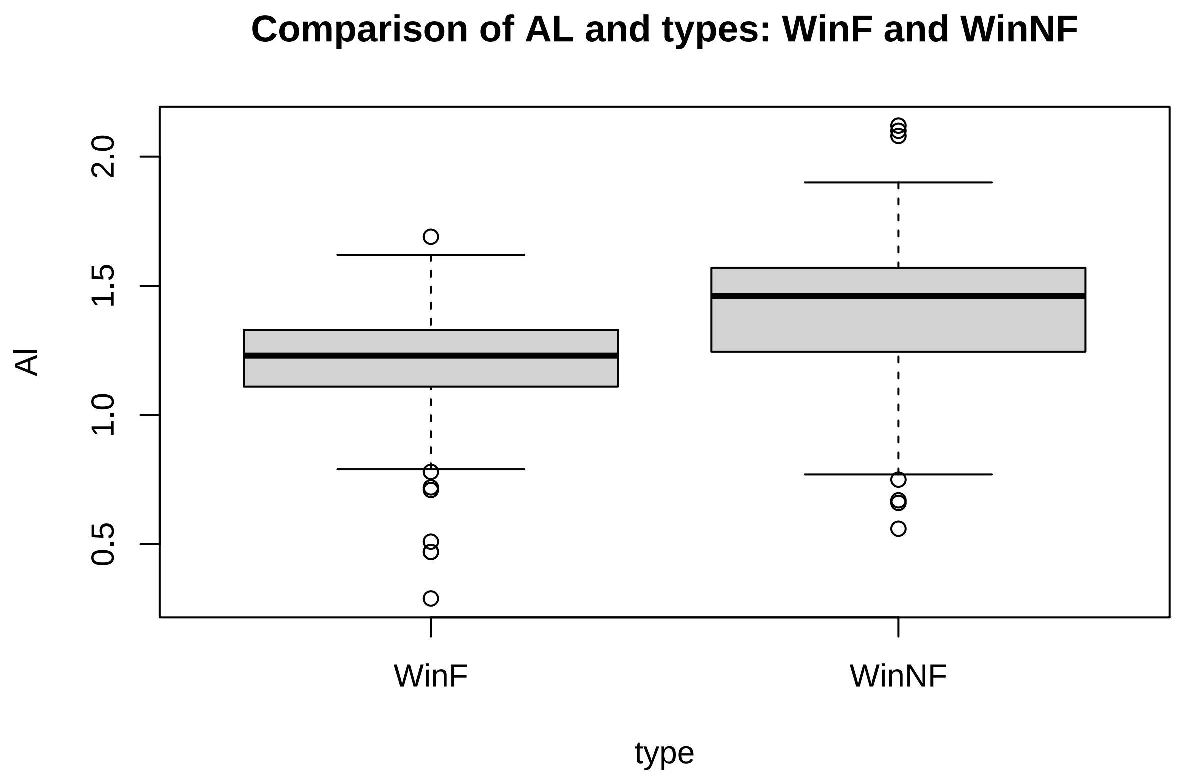


The Shapiro test performed on the residuals of our optimized model and training data indicates that they deviate from normality, as the p-value is less than 0.05. Conversely, when applying the Shapiro test to the residuals of our optimized model and testing data, we find that the residuals for the testing data follow a normal distribution, as the p-value (0.7865) is greater than 0.05. Utilizing the rainbow test on our optimized model, we find no evidence of violating the linearity assumption. The null hypothesis of a linear fit is not rejected, as the p-value (0.9892) is greater than 0.05. The test for multicollinearity results in GVIFs lower than 5 for all variables, indicating the absence of multicollinearity. According to the Durban-Watson test for autocorrelation, we can confidently assume that the residuals are not correlated with one another, given the p-value exceeding 0.05. The Breusch Pagan test for heteroskedasticity reveals a violation of the equal variance assumption in our optimized model, as the p-value is less than 0.05. Moreover, upon observing the Studentized Residuals plot, it becomes evident that there is an outlier present. This outlier is observed in our training data but not in our testing data.

Part (d)

```
glass_mutate <- glass %>%
  filter(type == "WinF" | type == "WinNF")

# Check 1 Independent of one another
boxplot(Al ~ type,
        data = glass_mutate,
        main = "Comparison of AL and types: WinF and WinNF")
```



```r
# Check 2: Normality
shapiro.test(glass_mutate$Al)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  glass_mutate$Al
## W = 0.97374, p-value = 0.006643
```

```r
# Check 3: Levene's test, The two groups have equal variance
var.test(Al ~ type,
         data = glass_mutate,
         conf.level = 0.9)
```

```
##
##  F test to compare two variances
##
## data:  Al by type
## F = 0.73628, num df = 69, denom df = 75, p-value = 0.1989
## alternative hypothesis: true ratio of variances is not equal to 1
## 90 percent confidence interval:
##  0.4991165 1.0901490
## sample estimates:
## ratio of variances
##          0.7362831
```

```
# Perform a non-parametric test: Since the distribution of Al is not normal, use a
# non-parametric test, such as the Wilcoxon-Mann-Whitney test, to compare the
# medians of the two groups
wilcox_result <- wilcox.test(Al ~ type,
                             data = glass_mutate,
                             var.equal = TRUE,
                             conf.level = 0.9)
wilcox_result
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  Al by type
## W = 1399, p-value = 7.875e-07
## alternative hypothesis: true location shift is not equal to 0
```

```
p_value <- wilcox_result$p.value
```

```
##
## The distributions of the two groups are significantly different at the 10% significance level.
```

The Shaprio test reveals a deviation from normal distribution in Al, as evidenced by a p-value below 0.05. Consequently, a t-test comparing the means of the two groups cannot be conducted due to the non-normality. However, the two groups are independent and exhibit variances that are not significantly dissimilar, as indicated by an F test p-value exceeding 0.05. Thus, it is appropriate to employ a non-parametric test like the Wilcoxon-Mann-Whitney test to compare the medians of the two groups. Employing a 10% level of significance, the distributions of the two groups, WinF and WinNF, are markedly distinct since the p-value for the Wilcoxon test is significantly below 0.05.

Part (e):

```
set.seed(123)
# Need 214 values
N <- 214

Na <- rnorm(n=N, mean = mean(glass$Na), sd(glass$Na))
Mg <- rnorm(n=N, mean = mean(glass$Mg), sd(glass$Mg))
Al <- rnorm(n=N, mean = mean(glass$Al), sd(glass$Al))
K <- rnorm(n=N, mean = mean(glass$K), sd(glass$K))
Ca <- rnorm(n=N, mean = mean(glass$Ca), sd(glass$Ca))
Ba <- rnorm(n=N, mean = mean(glass$Ba), sd(glass$Ba))
Fe <- rnorm(n=N, mean = mean(glass$Fe), sd(glass$Fe))


e <- rnorm(n=N, 0, 1)

# Saves coefficients from our model3 as variables
coef <- summary(final_model)$coefficients

Si <- e + coef[1] + coef[2]*Na + coef[3]*Mg + coef[4]*Al + coef[5]*K +
  coef[6]*Ca + coef[7]*Ba + coef[8]*Fe

simulated_glass <- cbind(Na, Mg, Al, K, Ca, Ba, Fe, Si)
simulated_glass <- as.data.frame(simulated_glass)

summary(Si)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   65.79   71.00   72.43   72.56   74.44   80.43
```

Using the mean and standard deviation of each observed variable, we generated simulated data for each variable. Given that our observed data is approximately normally distributed, we opted to employ the normal distribution method for generating the simulated data for all the independent variables.

Upon examination, we observed that the simulated Si values range from 65.79 to 80.43. The median value of our simulated Si data is 72.43, with a mean of 72.56. In comparison, the observed data has a median of 72.79 and a mean of 75.65. The interquartile range of the simulated Si values falls within 71.00 to 74.44.

```
set.seed(123)
# Simulated model using the technique
training <- simulated_glass$Si %>%
    createDataPartition(p = .75, list = FALSE)
train.data  <- simulated_glass[training, ]
test.data <- simulated_glass[-training, ]

sim_model<- lm(Si ~ Na + Mg + Al + K + Ca + Ba + Fe, data = train.data)
predictions <- glass_model3 %>% predict(test.data)
sim_model_results <- data.frame( R2 = R2(predictions, test.data$Si),
            RMSE = RMSE(predictions, test.data$Si),
            MAE = MAE(predictions, test.data$Si))

sim_model_results
```

```
##          R2     RMSE       MAE
## 1 0.8415506 1.073377 0.8511072
```

```
summary(sim_model)
```

```
##
## Call:
## lm(formula = Si ~ Na + Mg + Al + K + Ca + Ba + Fe, data = train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6069 -0.6098 -0.1213  0.7546  3.0007
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 99.01461    1.56699  63.188  < 2e-16 ***
## Na          -0.95595    0.10516  -9.091 4.62e-16 ***
## Mg          -0.98848    0.05688 -17.379  < 2e-16 ***
## Al          -0.99373    0.16071  -6.184 5.39e-09 ***
## K           -1.12794    0.11973  -9.421  < 2e-16 ***
## Ca          -0.97074    0.05444 -17.832  < 2e-16 ***
## Ba          -0.76646    0.17076  -4.489 1.40e-05 ***
## Fe           1.29268    0.85387   1.514    0.132
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.017 on 154 degrees of freedom
## Multiple R-squared:  0.8439, Adjusted R-squared:  0.8368
## F-statistic: 118.9 on 7 and 154 DF,  p-value: < 2.2e-16
```

Simulated Model Regression equation:

$$\begin{aligned} Si = 99.01 - (0.96 \cdot Na) - (0.99 \cdot Mg) - (0.99 \cdot Al) \\ - (1.13 \cdot K) - (0.97 \cdot Ca) - (0.77 \cdot Ba) + (1.29 \cdot Fe) \end{aligned}$$

This regression equation suggests that when none of the critical elements are present in the glass, 99.01% of the material will be Si. The negative slopes in the equation means that an increase of the presence of any other element by one percent will decrease the Si content of the glass by roughly one percent, which makes sense.

It is clearly to see that our simulated regression has completely different coefficients for iron where for every one percent of increase in iron, the percentage of materials that are silicon in the glass will increases by 1.3%. (Certain chemical properties?)

```
# Check for normality using shapiro test on residuals
sim_resid <- residuals(sim_model)
shapiro.test(sim_resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sim_resid
## W = 0.99387, p-value = 0.7324
```

```
# Check for Linearity using rainbow test
raintest(sim_model)
```

```
##
##  Rainbow test
##
## data:  sim_model
## Rain = 0.71802, df1 = 81, df2 = 73, p-value = 0.9267
```

```
# Check for multicollinearity
cat("Check for multicollinearity\n\n")
```

```
## Check for multicollinearity
```

```
vif(sim_model)
```

```
##       Na       Mg       Al        K       Ca       Ba       Fe
## 1.011810 1.022289 1.006379 1.007384 1.017936 1.020900 1.020869
```

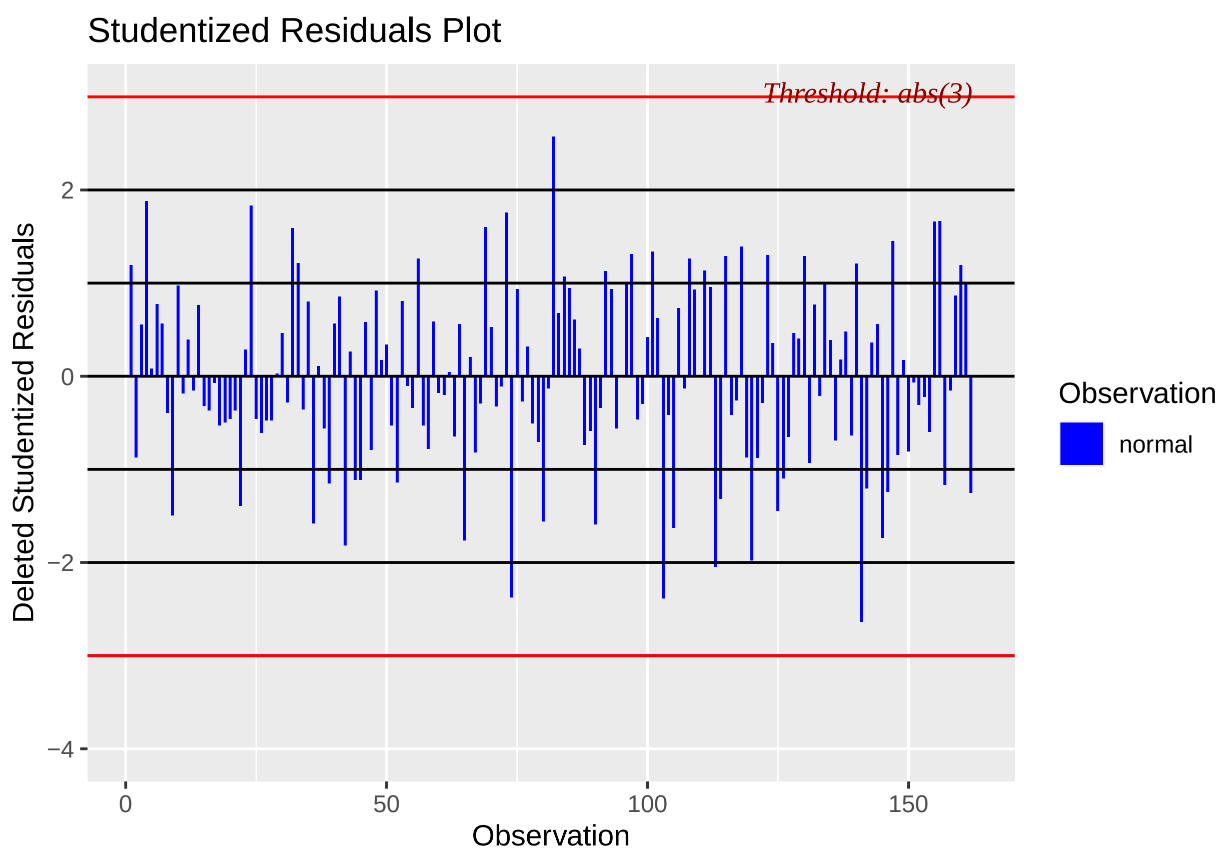
```
# Check for autocorrelation
dwtest(sim_model)
```

```
##
##  Durbin-Watson test
##
## data:  sim_model
## DW = 2.0209, p-value = 0.5619
## alternative hypothesis: true autocorrelation is greater than 0
```

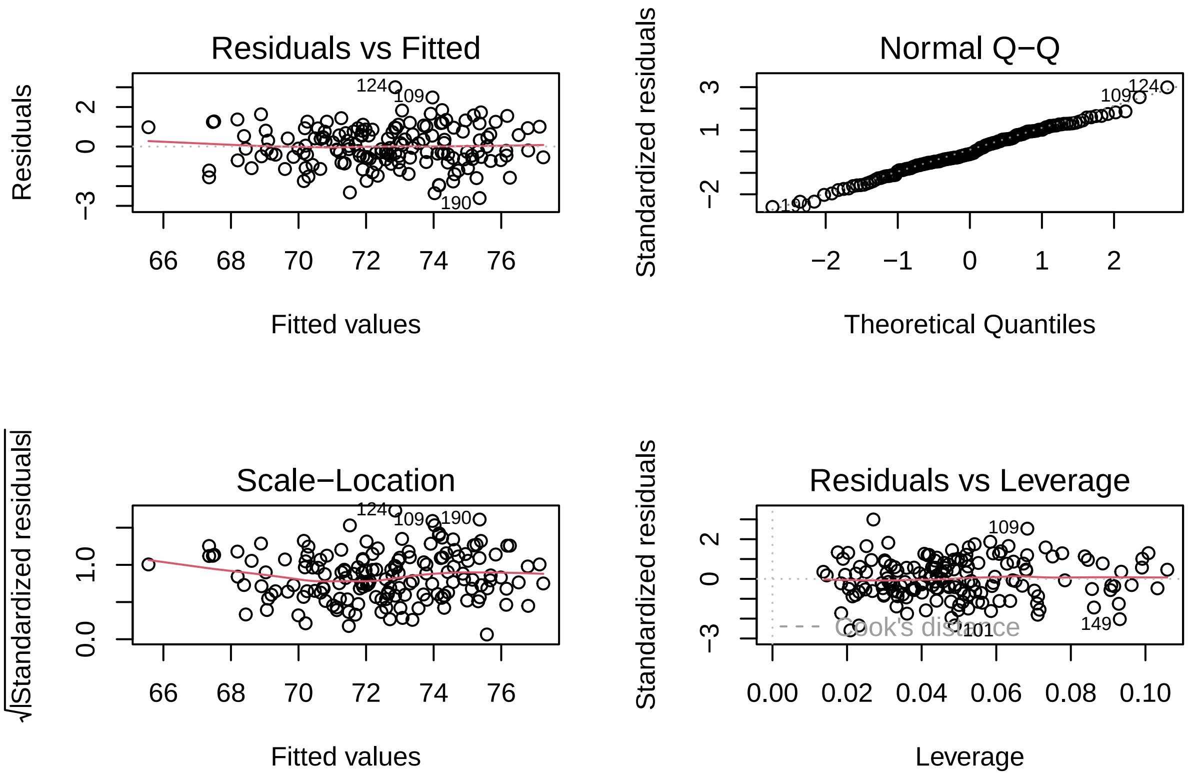
```
# Check for Heteroskedasticity
ols_test_breusch_pagan(sim_model)
```

```
##
##  Breusch Pagan Test for Heteroskedasticity
##  -----------------------------------------
##  Ho: the variance is constant
##  Ha: the variance is not constant
##
##             Data
##  ------------------------------
##  Response : Si
##  Variables: fitted values of Si
##
##        Test Summary
##  ---------------------------
##  DF            =    1
##  Chi2          =    1.343856
##  Prob > Chi2   =    0.246355
```

```
# Check for outliers
ols_plot_resid_stud(sim_model)
```



```
par(mfrow = c(2,2))
plot(sim_model)
```



The Shapiro test conducted on the residuals of our simulated model and simulated data indicates that they follow a normal distribution, as the p-value is greater than 0.05. By performing the rainbow test on our simulated model, we find no evidence of violating the assumption of linearity. The null hypothesis of a linear fit is not rejected, as the p-value is greater than 0.05. The test for multicollinearity yields GVIFs less than 5 for all variables, suggesting the absence of multicollinearity. According to the Durban-Watson test for autocorrelation, we can safely assume that the residuals are not correlated with each other, given the p-value greater than 0.05. The Breusch Pagan test for heteroskedasticity indicates that the equal variance assumption holds in our simulated model. Upon examination of the Studentized Residuals plot, there is no clear evidence of outliers present in the residuals of our simulated model.

Upon comparing the residuals of our optimized model to those of the simulated model, we observe that all assumptions remain consistent, except for the fact that our simulated data follows a normal distribution. This outcome aligns with expectations, as the simulated data was intentionally generated to adhere to a normal distribution.

Problem 3

Part (a)

We opted to select independent variables that exclusively comprised discrete or continuous numerical values, excluding factors such as the teacher's sex, type of school, or teaching level. Our decision was based on the belief that this variable selection, combined with 10-fold cross-validation, would yield the highest level of accuracy in our model.

10-fold Cross-Validation

```r
# Convert COMMIT to binary < median = 0, otherwise = 1.

new_Handout1 <- Handout1 %>%
  mutate(COMMIT = case_when(
    (COMMIT < median(COMMIT)) ~ 0,
    TRUE ~ 1))

# Select only the continuous numerical variables
new_Handout1 <- new_Handout1 %>%
  select(COMMIT, AGE, SALARY, CLASSSIZE, RESOURCES, AUTONOMY, CLIMATE, SUPPORT)


# Training and testing method
set.seed(123)
# Create training and testing data
index <- createDataPartition(new_Handout1$COMMIT,
                             p = 0.8,
                             list = FALSE,
                             times=1)


new_Handout1 <- as.data.frame(new_Handout1)

train  <- new_Handout1[index, ]
test <- new_Handout1[-index, ]

train$COMMIT[train$COMMIT==1] <- "yes"
train$COMMIT[train$COMMIT==0] <- "no"

test$COMMIT[test$COMMIT==1] <- "yes"
test$COMMIT[test$COMMIT==0] <- "no"


# Convert outcome variable to factor for each data frame
train$COMMIT <- as.factor(train$COMMIT)
test$COMMIT <- as.factor(test$COMMIT)

# 10-fold Cross-validation method
ctrlspecs <- trainControl(method="cv",
                          number=10,
                          savePredictions="all",
                          classProbs=TRUE)

set.seed(1234)
cvmodel <- train(COMMIT ~ .,
                 data=train,
                 method="glm",
                 family = "binomial",
                 trControl=ctrlspecs)

# Model summary for 10-fold Cross-validation method
```

```r
summary(cvmodel)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.06170  -0.90585   0.08378   0.93637   1.94307
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.38779    5.07406  -2.047   0.0406 *
## AGE           0.14086    0.09033   1.559   0.1189
## SALARY        0.11065    0.09566   1.157   0.2474
## CLASSSIZE    -0.14105    0.06897  -2.045   0.0408 *
## RESOURCES     0.18733    0.13923   1.345   0.1785
## AUTONOMY      0.30463    0.12547   2.428   0.0152 *
## CLIMATE       0.21419    0.09829   2.179   0.0293 *
## SUPPORT      -0.19250    0.13418  -1.435   0.1514
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 166.36  on 119  degrees of freedom
## Residual deviance: 130.47  on 112  degrees of freedom
## AIC: 146.47
##
## Number of Fisher Scoring iterations: 4
```

```r
# Model for 10-fold Cross-validation method
print(cvmodel)
```

```
## Generalized Linear Model
##
## 120 samples
##   7 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.675     0.35
```

```r
# Important variables for 10-fold Cross-validation method
varImp(cvmodel)
```

```
## glm variable importance
```

```
##
##           Overall
## AUTONOMY   100.00
## CLIMATE     80.43
## CLASSSIZE   69.90
## AGE         31.67
## SUPPORT     21.86
## RESOURCES   14.85
## SALARY       0.00
```

Model Accuracy

```
# Predict outcome using model from training data based on testing data
predictions <- predict(cvmodel, newdata=test)

# Create confusion matrix to assess model fit/performance on test data
con_matx <- confusionMatrix(data=predictions, test$COMMIT)
con_matx
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction no yes
##        no  13   4
##        yes  2  11
##
##                Accuracy : 0.8
##                  95% CI : (0.6143, 0.9229)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0007155
##
##                   Kappa : 0.6
##
##  Mcnemar's Test P-Value : 0.6830914
##
##             Sensitivity : 0.8667
##             Specificity : 0.7333
##          Pos Pred Value : 0.7647
##          Neg Pred Value : 0.8462
##              Prevalence : 0.5000
##          Detection Rate : 0.4333
##    Detection Prevalence : 0.5667
##       Balanced Accuracy : 0.8000
##
##        'Positive' Class : no
##
```

After examining the confusion matrix, it becomes apparent that our model attained an accuracy rate of 80% (95% CI: 0.61-0.92) when tested on the data. As the p-value exceeds 0.05, this suggests that our model exhibits a statistically significant level of accuracy.

Part (b)

Our objective is to optimize the cost function in order to identify the most economical approach for fulfilling the Christmas order. This entails striving for the lowest cost per day to achieve maximum cost efficiency. By minimizing expenses at each factory, we can ensure a cost-effective process for meeting the demands of the Christmas order.

```r
factories <- c("Factory A", "Factory B", "Factory C")
toys <- c("Cars", "Animals", "Robots")

# Define the coefficients of the objective function
costs <- c(1000, 2100, 1500)

# Define the constraint matrix
constraint_matrix <- matrix(c(30, 20, 30,
                              40, 50, 10,
                              50, 40, 15), nrow = 3, byrow = TRUE)

# Define the right-hand side of the constraints
constraint_limits <- c(5000, 3000, 2500)

# Set the direction of optimization (minimization)
constraint_directions <- c(">=", ">=", ">=")
direction <- "min"

# Solve the linear programming problem
min_solution <- lp(direction = direction,
                   objective.in = costs,
                   const.mat = constraint_matrix,
                   const.dir = constraint_directions,
                   const.rhs = constraint_limits,
                   compute.sens=TRUE)

# Check if a solution was found
if (min_solution$status == 0) {
  # Print the optimal solution
  cat("The optimal solution is:\n")
  cat("Factory A:",min_solution$solution[1], "days","\n")
  cat("Factory B:",min_solution$solution[2], "days","\n")
  cat("Factory C:",min_solution$solution[3], "days","\n\n")
  # Print the minimum cost
  cat("The value of the objective function at the optimal solution is:\n")
  cat("Minimum cost: $",min_solution$objval)
} else {
  # No feasible solution found
  print("No feasible solution found.")
}
```

```
## The optimal solution is:
## Factory A: 166.6667 days
## Factory B: 0 days
## Factory C: 0 days
##
## The value of the objective function at the optimal solution is:
```

```
## Minimum cost: $ 166666.7
```

The most efficient approach entails running Factory A for approximately 166.67 days, resulting in a minimal cost of $166,666.70, whereas Factory B and Factory C remain non-operational. By adopting this strategy, the company can achieve the most cost-effective outcome.

When considering the optimal solution, it is important to analyze the cost implications of running the factories for different durations. In this scenario, operating Factory for approximately 166.67 days leads to the minimum overall cost. By halting the operations of Factory B and Factory C, the company can avoid additional expenses associated with their functioning.

```
## The summary of the results:
```

```
##             Minimum operating days Minimum costs
## 1 Factory A               166.6667      166666.7
## 2 Factory B                 0.0000           0.0
## 3 Factory C                 0.0000           0.0
```

Part (c)

```r
# Weibull statistics
# lambda is scale while K is shape
compute_weibull_stats <- function(shape, scale) {

  # Compute the mean
  mean_value <- scale * gamma(1 + (1/shape))

  # Compute the median
  median_value <- scale * (log(2)^(1/shape))

  # Compute the mode
  if (shape > 1)
    mode_value <- scale * ((shape - 1) / shape)^(1/shape)
  else
    mode_value <- 0

  # Compute the variance
  variance_value <- scale^2 * (gamma(1 + (2/shape)) - (gamma(1 + (1/shape)))^2)

  # Return the computed statistics as a named list
  return(list(mean = mean_value,
              median = median_value,
              mode = mode_value,
              variance = variance_value))
}

# Example usage
scale_param <- 6
shape_param <- 1

stats <- compute_weibull_stats(shape_param, scale_param)
```

```r
# Accessing the computed statistics
mean_value <- stats$mean
median_value <- stats$median
mode_value <- stats$mode
variance_value <- stats$variance
```

When $\lambda = 6$ and $k = 1$ then the Weibull statistics are:
$Mean = 6$
$Median = 4.1588831$
$Mode = 0$
$Variance = 36$

Maximum likelihood estimator (MLE) of the parameters from the Weibull distribution

Probability Density Function ($f(x; \lambda, k)$) for the Weibull distribution:

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^{k}}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

The likelihood function ($L_{\hat{x}}(\lambda, k)$):

$$\begin{aligned} L_{\hat{x}}(\lambda,k) &= \prod_{i=1}^{n} \frac{k}{\lambda}\left(\frac{x_i}{\lambda}\right)^{k-1} e^{-\left(\frac{x_i}{\lambda}\right)^{k}} \\ &= \left(\frac{k}{\lambda}\right)^{n}\left(\frac{1}{\lambda^{k-1}}\right)^{n}\left(\prod_{i=1}^{n} x_i^{k-1}\right)\left(e^{-\sum\limits_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^{k}}\right) \\ &= \frac{k^n}{\lambda^{nk}} e^{-\sum\limits_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^{k}} \prod_{i=1}^{n} x_i^{k-1} \end{aligned}$$

The log-likelihood function ($\ln L_{\hat{x}}(\lambda, k)$):

$$\ln L_{\hat{x}}(\lambda,k) = n\ln(k) - nk\ln(\lambda) - \sum_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^{k} + (k-1)\sum_{i=1}^{n}\ln x_i$$

Partial derivative of the log-likelihood function with respect to λ:

$$\frac{\partial \ln L_{\hat{x}}(\lambda,k)}{\partial\lambda} = -\frac{nk}{\lambda} + k\sum_{i=1}^{n}\frac{x_i^k}{\lambda^{k+1}}$$

Solving for the desired parameter of λ by setting the partial derivative equal to zero:

$$\begin{aligned}
\frac{\partial \ln L_{\hat{x}}(\lambda, k)}{\partial \lambda} &= 0 \\
-\frac{nk}{\lambda} + k\sum_{i=1}^{n} \frac{x_i^k}{\lambda^{k+1}} &= 0 \\
-\frac{nk}{\lambda} + \frac{k}{\lambda}\sum_{i=1}^{n} \left(\frac{x_i}{\lambda}\right)^k &= 0 \\
-n + \sum_{i=1}^{n} \frac{x_i^k}{\lambda^k} &= 0 \\
\frac{1}{\lambda^k}\sum_{i=1}^{n} x_i^k &= n \\
\frac{1}{n}\sum_{i=1}^{n} x_i^k &= \lambda^k
\end{aligned}$$

Therefore the estimator $\hat{\lambda}$ is:

$$\hat{\lambda} = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^k\right)^{\frac{1}{k}}$$

Plugging in $\hat{\lambda}$ into the log-likelihood function ($\ln L_{\hat{x}}(\lambda, k)$) and then differentiating with respect to k in order to find the estimator $\hat{k}$:

$$\begin{aligned}
\frac{\partial \ln L_{\hat{x}}(\lambda, k)}{\partial k} &= \frac{\partial}{\partial k}\left[n\ln k - nk\ln\lambda - \sum_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^k + (k-1)\sum_{i=1}^{n}\ln x_i\right] \\
&= \frac{\partial}{\partial k}\left[n\ln k - nk\ln\left[\left(\frac{1}{n}\sum_{i=1}^{n} x_i^k\right)^{\frac{1}{k}}\right] - \frac{\sum\limits_{i=1}^{n} x_i^k}{\left[\left(\frac{1}{n}\sum\limits_{i=1}^{n} x_i^k\right)^{\frac{1}{k}}\right]^k} + (k-1)\sum_{i=1}^{n}\ln x_i\right] \\
&= \frac{\partial}{\partial k}\left[n\ln k - \frac{nk}{k}\ln\left(\frac{1}{n}\sum_{i=1}^{n} x_i^k\right) - \frac{\sum\limits_{i=1}^{n} x_i^k}{\left(\frac{1}{n}\sum\limits_{i=1}^{n} x_i^k\right)} + (k-1)\sum_{i=1}^{n}\ln x_i\right] \\
&= \frac{\partial}{\partial k}\left[n\ln k - n\ln\left(\frac{1}{n}\sum_{i=1}^{n} x_i^k\right) - n + (k-1)\sum_{i=1}^{n}\ln x_i\right] \\
&= \frac{n}{k} - \left(\frac{n\sum\limits_{i=1}^{n} x_i^k\ln x_i}{\sum\limits_{i=1}^{n} x_i^k}\right) + \sum_{i=1}^{n}\ln x_i \\
&= \frac{1}{k} - \left(\frac{\sum\limits_{i=1}^{n} x_i^k\ln x_i}{\sum\limits_{i=1}^{n} x_i^k}\right) + \frac{1}{n}\sum_{i=1}^{n}\ln x_i
\end{aligned}$$

Solving for the desired parameter of k by setting the partial derivative equal to zero:

$$\begin{aligned}
\frac{\partial \ln L_{\hat{x}}(\lambda, k)}{\partial k} &= 0 \\
\frac{1}{k} - \left(\frac{\sum\limits_{i=1}^{n} x_i^k \ln x_i}{\sum\limits_{i=1}^{n} x_i^k} \right) + \frac{1}{n} \sum_{i=1}^{n} \ln x_i &= 0 \\
\left(\frac{\sum\limits_{i=1}^{n} x_i^k \ln x_i}{\sum\limits_{i=1}^{n} x_i^k} \right) - \frac{1}{n} \sum_{i=1}^{n} \ln x_i &= \frac{1}{k}
\end{aligned}$$

Therefore the estimator $\hat{k}$ is:

$$\hat{k} = \left[\frac{\sum\limits_{i=1}^{n} x_i^k \ln x_i}{\sum\limits_{i=1}^{n} x_i^k} - \frac{1}{n} \sum_{i=1}^{n} \ln x_i \right]^{-1}$$

The definition of $\hat{k}$ provided here is implicit, as determining the value of k typically requires numerical methods for solution.